

# Initialization Algorithms For Convolutional Network Coding

Maxim Lvov and Haim H. Permuter

## Abstract

We present algorithms for initializing a convolutional network coding scheme in networks that may contain cycles. An initialization process is needed if the network is unknown or if local encoding kernels are chosen randomly. During the initialization process every source node transmits basis vectors and every sink node measures the impulse response of the network. The impulse response is then used to find a relationship between the transmitted and the received symbols, which is needed for a decoding algorithm and to find the set of all achievable rates. Unlike acyclic networks, for which it is enough to transmit basis vectors one after another, the initialization of cyclic networks is more involved, as pilot symbols interfere with each other and the impulse response is of infinite duration.

**Keywords:** Cayley-Hamilton Theorem, Convolutional network coding, Cyclic networks, Linear network coding, System identification.

## I. INTRODUCTION

Network coding is a technique that is used to increase a network's throughput. The idea behind this coding scheme is that the relay nodes transmit functions of the received symbols on their output links, rather than simply routing them. Ahlswede *et al.* [1] showed that for a one source, multicast, acyclic network, the maximal network's throughput is

The work of M. Lvov and H. Permuter was supported by the Israel Science Foundation (grant no. 684/11), by the ERC starting grant and by the Israeli Ministry of Defense.

This paper will be presented in part at the 2014 International Symposium on Network Coding, Aalborg, Denmark.

M. Lvov and H. H. Permuter are with the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, 84105, Beer-Sheva, Israel (email: maxlvo55@gmail.com; haimp@bgu.ac.il).

equal to the minimum cut between the source and any sink node. They also showed that for some networks, the ordinary routing scheme cannot achieve the min-cut bound, although a network coding scheme can. For cyclic networks, a Convolution Network Coding (CNC) scheme was presented by Li *et al.* [2], and the existence of an optimal CNC code (one that achieves the min-cut bound given in [1]) was proved by Koetter and Médard [3]. Since then, much work has been devoted to constructing codes for cyclic networks [3]–[7], but all these code-construction algorithms share one major drawback; they all need to know in advance the network topology. In particular, if the network is large, it might be difficult to learn the exact network structure.

A randomized linear network coding approach was presented by Ho *et al* [8]. They showed that for a cyclic network, all sink nodes will be able, with high probability, to decode the symbols sent by the source nodes, provided that the transmission rates of all sources satisfy the Min-Cut Max-Flow condition and that the local encoding kernels are chosen randomly from a large enough field. The Min-Cut Max-Flow condition states that for every subset  $\mathcal{A}$  of source nodes, the sum of source rates  $\sum_{s \in \mathcal{A}} R_s$  must be less than or equal to the minimum cut capacity between every sink node and  $\mathcal{A}$ .

This result makes random linear encoding extremely useful when the network is dynamic and no central authority for assigning encoding kernels exists. The local encoding kernels can be chosen randomly from some large enough field and, with high probability, this will lead to a network that allows source nodes to transmit symbols at high rates, thereby enabling all sink nodes to decode the sent symbols. This outcome, however, requires that the source nodes know the capacity region and that the sink nodes know a decoding algorithm. If the network structure or the local encoding kernels are not known, an initialization process is needed.

In this paper, we present two initialization algorithms that find a decoding scheme for the sink nodes and one algorithm that finds the capacity region for the source nodes. The decoding scheme is found by sending pilot basis vectors and measuring the impulse response of the network, a method analogous to the one given in [9, p. 448] for acyclic networks. Although the impulse response of the network can be of infinite duration, our

algorithms find a decoding scheme using only the initial values of the impulse response. In the first algorithm, we transmit basis vectors and measure the impulse response of the network under the assumption that the initial symbols sent on the network are zeros. In the second algorithm, we assume that neither the initial symbols are zeros nor that it is possible to clear all these symbols at once. Our algorithms do not require any additional headers to be transmitted. This simplifies the design of the relay nodes, since they do not operate differently during and after the initialization process. The method for finding the capacity region is based on the fact that the connection between the source and the sink nodes is possible if the transfer matrix is of full rank [3].

A randomized initialization of convolutional network codes was introduced by Guo *et al* [10]. Their method used a time-variant decoding algorithm proposed in [11] to decode the transmitted symbols. By that method, one can decode all of the transmitted symbols up to time  $n$  by using the first  $n + L$  terms of the network's impulse response, where  $L$  is the decoding delay. Our results can improve their algorithms since we have developed a method to find the full impulse response (by expanding the global encoding kernels, found in the initialization process, into power series) from a finite set of its initial values. After finding the global encoding kernels, both time-variant decoding [11] presented by Guo *et al*, and the sequential decoding algorithm [7] presented by Erez and Feder can be used.

Methods for identifying an unknown linear time-invariant (LTI) system from its impulse response are well known from control theory. In particular, these methods are used to find a state space representation of the system, i.e. to find the matrices **A**, **B**, **C** and **D** such that the following state equations will satisfy the input-output relationship of the system:

$$\begin{aligned}\mathbf{x}[n + 1] &= \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n], \\ \mathbf{y}[n] &= \mathbf{C}\mathbf{x}[n] + \mathbf{D}\mathbf{u}[n],\end{aligned}\tag{1}$$

where  $\mathbf{u}[n]$  is the input vector,  $\mathbf{y}[n]$  is the output vector and  $\mathbf{x}[n]$  is the state vector. Usually the state space representation obtained by these methods is an approximate one and is based on statistical methods [12]. After such a representation is found, the transfer

function can be found by applying the  $Z$ -transform on (1):

$$\mathbf{H}(z) = \mathbf{C} \cdot \text{adj}(z\mathbf{I} - \mathbf{A}) \cdot \mathbf{B}/P_{\mathbf{A}}(z) + \mathbf{D}. \quad (2)$$

Here,  $P_{\mathbf{A}}(z) = \det(z\mathbf{I} - \mathbf{A})$  is the characteristic polynomial of  $\mathbf{A}$  and  $\mathbf{H}(z)$  is the transfer function of the system. A cyclic network with a convolutional network coding scheme can also be described by a state space representation, as was introduced by Fragouli and Soljanin [13].

System identification is closely related to the initialization process we show here. In both cases we have an unknown LTI system, for which we want to find the input-output relationship without learning the exact structure of the system, but only by sending pilot input vectors. However, our motivation for finding this input-output relationship differs from that usually cited in control theory, where we tend to look for the input sequence in order to obtain the desired output sequence. In our case, we need the input-output relationship to be able to decode the transmitted symbols and to find the capacity region for all source nodes. There are also other differences between system identification and our initialization process, such as the fact that LTI systems usually work in the field of real or complex numbers, while the networks we work with use finite fields.

One of the methods to find an input-output relationship of a deterministic LTI system from its impulse response is described by the Ho-Kalmans Method [12, p. 142]. Using that method, we first need to measure the impulse response  $\{\mathbf{G}[n]\}_{n=1}^{k+l}$  (where  $k, l$  are any numbers that are greater than the order of the system) and construct the Hankel matrix

$$\mathbf{H}_{k,l} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 & \cdots & \mathbf{G}_l \\ \mathbf{G}_2 & \mathbf{G}_3 & \mathbf{G}_4 & \cdots & \mathbf{G}_{l+1} \\ \mathbf{G}_3 & \mathbf{G}_4 & \mathbf{G}_5 & \cdots & \mathbf{G}_{l+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_k & \mathbf{G}_{k+1} & \mathbf{G}_{k+2} & \cdots & \mathbf{G}_{l+k} \end{bmatrix}. \quad (3)$$

Using a singular value decomposition (SVD) of  $\mathbf{H}_{k,l}$ , a minimal realization of the system is constructed as described in [12], from which a transfer function is found. This method, however, assumes that the field over which linear combinations are performed is the set of

real or complex numbers. In our case, the field is finite and no SVD operation is defined. A method for system realization from its Hankel matrix is described in [14, p. 498], but it requires us to know the rank of Hankel matrices of higher orders (for larger  $k, l$ ).

The transfer function of an LTI system can be found if one knows the characteristic polynomial  $P_{\mathbf{A}}(z)$  of the matrix  $\mathbf{A}$ . One can pass the output of the system through a finite impulse response (FIR) filter with a transfer function  $P_{\mathbf{A}}(z)$  such that the total transfer function of the cascaded system would be

$$\mathbf{H}(z)P_{\mathbf{A}}(z) = \mathbf{C} \cdot \text{adj}(z\mathbf{I} - \mathbf{A}) \cdot \mathbf{B}. \quad (4)$$

The transfer function in (4) is a polynomial in  $z$  and, hence, can be obtained by sending basis vectors and measuring the finite impulse response. A method to find the characteristic polynomial  $P_{\mathbf{A}}(z)$  from the diagonal minors of the Hankel matrix was introduced by Sreeram in [15]. However, this method requires us to know the order of the system, i.e. the dimension of the state vector in its minimal realization, which is not usually known a priori when we consider unknown networks. In the methods we present only the number of edges and the maximal transmission rate for every source (or an upper bound for each of them) are needed.

The paper is divided into seven sections. In Section II we outline notations and define the problem. In Section III we present two algorithms for network initialization and one for finding the capacity region of the network. In Sections IV, V and VI we explain why these algorithms work, one algorithm per section. Section VII concludes the paper. In Appendix A we show examples for applying the algorithms. In Appendix B we give the proofs for all the theorems and lemmas.

## II. NOTATIONS AND PROBLEM DEFINITION

We represent a communication network by a directed graph  $G = (\mathcal{V}, \mathcal{E})$  where  $\mathcal{V}$  is the set of nodes and  $\mathcal{E}$  is the set of edges. Each edge represents a noiseless directed link that can transmit one symbol per unit time, where the symbols are scalars from some field  $\mathbb{F}$ . We assume every link has a unit time delay between consequent symbol transmissions and transmissions on all links are synchronized.

We denote by  $\mathcal{S}$  the set of all source nodes and by  $\mathcal{D}$  the set of all sink nodes. Every source node  $s \in \mathcal{S}$  transmits  $R_s$  symbols per unit time. Every sink node wants to receive all the symbols sent by all the source nodes. For every edge  $e \in \mathcal{E}$ , we say that  $u = \text{head}(e)$  and  $v = \text{tail}(e)$  if  $u, v \in \mathcal{V}$  and  $e$  is from  $v$  to  $u$ . We denote by  $\text{In}(u) = \{e \in \mathcal{E} : u = \text{head}(e)\}$  and  $\text{Out}(u) = \{e \in \mathcal{E} : u = \text{tail}(e)\}$ . The symbol that is sent on the edge  $e$  at time  $n \in \mathbb{Z}$  is denoted by  $x_e[n]$ . We denote vectors or sequences of vectors by lowercase bold letters, while matrices are denoted by bold capital letters. We assume there is a CNC scheme in the network, so that the symbol sent on a link  $i \in \text{Out}(j)$  is a linear combination of the symbols received and generated by the node  $j$  in the previous time slot. This relationship can be written as

$$x_i[n+1] = \sum_{e \in \text{In}(j)} a_{i,e} x_e[n] + \sum_{k=1}^{R_j} b_{i,k} u_{j,k}[n], \quad \forall i \in \mathcal{E}, \forall n \geq 0, \quad (5)$$

where  $u_{j,k}[n]$  is the  $k$ 'th symbol generated by node  $j$  (if  $j \in \mathcal{S}$ ) at time  $n$ , and  $\{a_{i,e}, b_{i,k}\}$  are the *local encoding kernels* for node  $j$  that were chosen in advance (probably randomly). By letting  $x_i[n]$  depend only on the previously received symbols, we avoid the problem described in [16] by Cai and Guo, when the convolutional code is not well defined in a cyclic network. If the network has a reset option that clears all the sent symbols in the network, we can assume that the initial network state is zero:

$$x_i[0] = 0, \quad \forall i \in \mathcal{E}.$$

**Example 1** As an example, we consider the network in Fig.1. There is one source node  $s_1$  and one sink node  $d_1$ . By the Min-Cut Max-Flow Theorem the rate  $R_{s_1} = 1$  is achievable, and the network state equations can be written in the next form:

$$\begin{pmatrix} x_1[n+1] \\ x_2[n+1] \\ x_3[n+1] \\ x_4[n+1] \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & \alpha_{1,4} \\ \alpha_{2,1} & 0 & \alpha_{2,3} & 0 \\ 0 & 0 & 0 & \alpha_{3,4} \\ 0 & \alpha_{4,2} & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1[n] \\ x_2[n] \\ x_3[n] \\ x_4[n] \end{pmatrix} + \mathbf{B}_{s_1} \cdot \mathbf{u}_{s_1}[n]. \quad (6)$$

If the rate  $R_{s_1}$  is set to 1 then  $u_{s_1}[n]$  is a scalar sequence and  $\mathbf{B}_{s_1} = (b_{1,1}, 0, b_{3,1}, 0)^T$  is

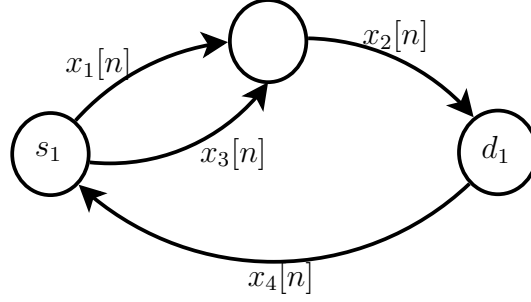


Fig. 1. Network with one source node, one sink node and one relay node.

a  $4 \times 1$  matrix over  $\mathbb{F}$ .

Note that we have restricted ourselves to the case where local encoding kernels are scalars, while in the general case they can be rational power series in the time shift operator [9, p. 492]. This, however, is not a major restriction, since one can achieve the capacity region without rational local encoding kernels if the field one works with is large enough [9, p. 502]. Nevertheless, we treat separately network codes with rational power series encoding kernels at the end of section III.

We define a time shift operator  $z$  acting on a sequence (of scalars or vectors)  $\{c[n]\}_{n \in \mathbb{Z}}$  as follows:

$$(z^k c)[n] = c[n + k], \forall k, n \in \mathbb{Z}. \quad (7)$$

Let  $P(t) = \sum_{k=0}^M a_k t^k$  be a polynomial in  $t$  with coefficients from the field  $\mathbb{F}$ . We define the operator  $P(z)$  as follows:

$$(P(z)c)[n] = \sum_{k=0}^M a_k c[n + k], \forall k, n \in \mathbb{Z}. \quad (8)$$

Finally, the coefficients  $\{a_k\}$  of  $P(t)$  can also be  $m \times k$  matrices over the field  $\mathbb{F}$ . In that case, the sequence  $\{c[n]\}_{n \in \mathbb{Z}}$  in (8) should be a sequence of  $k \times 1$  vectors. In order to avoid ambiguity, we will not use Z transforms of sequences and the symbol  $z$  will appear only as a time shift operator.

Let  $\mathbf{x}[n]$  be the column vector of size  $|\mathcal{E}|$  consisting of all symbols  $\{x_e[n]\}_{e \in \mathcal{E}}$  organized in some order. We define the *input sequence*  $\mathbf{u}[n] = \left( \mathbf{u}_{s_1}^T[n], \dots, \mathbf{u}_{s_{|\mathcal{S}|}}^T[n] \right)^T$  where  $\mathbf{u}_{s_i}[n] =$

$(u_{s_i,1}[n], \dots, u_{s_i,R_{s_i}}[n])^T$  is the *input sequence of source  $s_i$* , which is a sequence of vectors sent by source  $s_i$ . The dimension of the column vector  $\mathbf{u}[n]$  is  $m = \sum_{s \in \mathcal{S}} R_s$ . We assume that  $\mathbf{u}[n] = \mathbf{0}$  for  $n < 0$ . For every sink node  $d$ , we let  $\mathbf{y}_d[n]$  be a column vector consisting of all received symbols  $\{x_e[n] : e \in \text{In}(d)\}$  and the symbols generated by  $d, \{u_{d,k}[n]\}_{k=1}^{R_d}$  if  $d$  is also a source node, again organized in some order. The sequence  $\{\mathbf{y}_d[n]\}_{n \in \mathbb{Z}}$  will be called the *output sequence* of the sink node  $d$ , and the dimension of every vector in that sequence is  $l_d = R_d + |\text{In}(d)|$ . We assume also that  $\mathbf{y}_d[n] = \mathbf{0}$  for  $n < 0$ .

**Example 2** The shuttle network shown in Fig. 2 is used as an example. The nodes  $s_1, s_2$  are both source and sink nodes, and have the same transmission rates  $R_{s_1} = R_{s_2} = 1$ . The state vector is  $\mathbf{x}[n] = (x_1[n], x_2[n], \dots, x_8[n])^T$ , the input sequence is  $\mathbf{u}[n] = (u_{s_1,1}[n], u_{s_2,1}[n])^T$  ( $m = 2$ ), and the output sequences are  $\mathbf{y}_{s_1}[n] = (x_6[n], u_{s_1,1}[n])^T$  and  $\mathbf{y}_{s_2}[n] = (x_7[n], u_{s_2,1}[n])^T$ . Both  $l_{s_1}$  and  $l_{s_2}$  are equal to 2.

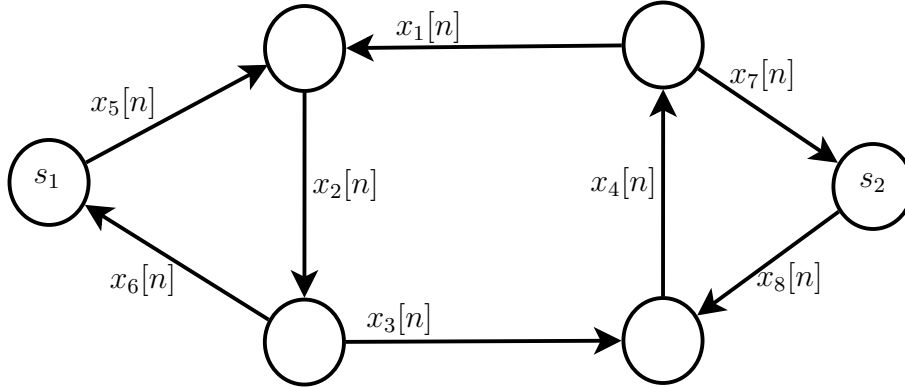


Fig. 2. Shuttle network with two users and 4 relay nodes.

We assume that either the network topology or the local encoding kernels or both are not known to any node a priori. We are interested in finding the network transfer matrix, or a way to decode the sent symbols  $\{\mathbf{u}[n]\}$  from the received symbols  $\{\mathbf{y}_d[n]\}$  at every sink node  $d$ , probably with some delay. This transfer function is obtained in our algorithms by sending pilot symbols and measuring the impulse response of the network. Even though we assume the network is unknown, our algorithms need all source and sink nodes to know some parameters of the network before the initialization process starts.



These parameters can be shared by some distribution protocol or assumed to be known a priori. The parameters are:

- The set of source nodes  $\mathcal{S}$  and their transmission rates  $\{R_s\}_{s \in \mathcal{S}}$  (or an upper bound for every rate),
- The number of edges in the network or an upper bound for it, which will be called  $N$ .

We next define the *achievable rates* for the network with specific local encoding kernels as the transmission rates  $(R_s)_{s \in \mathcal{S}}$  of all source nodes that will allow every sink node  $d$  to decode the vectors  $\{\mathbf{u}[n]\}_{n=0}^{n_0}$  from the vectors  $\{\mathbf{y}_d[n]\}_{n=0}^{n_0+\delta_d}$  (where  $\delta_d \geq 0$  represents the decoding delay, and is independent of  $n_0$ ) for all  $n_0 \in \mathbb{N}$ . The *capacity region* is defined as the set of all achievable rates.

In the definitions above, we do not restrict the sink nodes to any decoding method, even if these methods use the knowledge of the network topology and the local encoding kernels at every node. We do, however, restrict the network to have a CNC scheme with the chosen local encoding kernels. This restriction is not of great importance, since a CNC scheme with randomly chosen local encoding kernels can reach the capacity region given by the Min-Cut Max-Flow condition.

Before the initialization process starts, a transmission rate for every source node should be chosen. If an achievable rate for a specific source node is not known, it is preferable to set its rate to  $R_s = |\text{Out}(s)|$ . Algorithm 3, presented in the next section, can then be used to find achievable rates for this source. The source node  $s$  can then reduce its rate  $R_s$  to an achievable one by sending zeros on some of its input sequences  $\{u_{s,1}, \dots, u_{s,|\text{Out}(s)|}\}$ . In that case, we call the rates  $(R'_s)_{s \in \mathcal{S}}$  *achievable for a sink node  $d$*  if that sink node can decode the input sequence  $\mathbf{u}$  from the output  $\mathbf{y}_d$  when every source node  $s$  transmits symbols on  $R'_s$  out of its input sequences, and zeros on the rest of the  $(R_s - R'_s)$  input sequences. Note that the rates  $(R'_s)_{s \in \mathcal{S}}$  are achievable if they are achievable for every sink node.

**Example 3** Recall the network from Fig. 1 that was considered in Example 1. If the network topology and the capacity region are not known to  $s_1$ , the rate  $R_{s_1}$  can be set

to 2 as there are two outgoing links from the source node. In that case,  $\mathbf{u}_{s_1}[n]$  will be taken as a  $2 \times 1$  vector sequence and  $\mathbf{B}_{s_1}$  as a  $4 \times 2$  matrix:

$$\mathbf{B}_{s_1} = \begin{pmatrix} b_{1,1} & b_{1,2} \\ 0 & 0 \\ b_{3,1} & b_{3,2} \\ 0 & 0 \end{pmatrix}, \quad \mathbf{u}_{s_1}[n] = \begin{pmatrix} u_{s_1,1}[n] \\ u_{s_1,2}[n] \end{pmatrix}. \quad (9)$$

After finding the capacity region, the rate can be reduced to an achievable one by sending zeros on one of the input sequences

$$u_{s_1,1}[n] = 0, \forall n \in \mathbb{N} \quad \text{or} \quad u_{s_1,2}[n] = 0, \forall n \in \mathbb{N}. \quad (10)$$

### III. THE INITIALIZATION ALGORITHMS

In this section, we present two initialization algorithms that find a decoding scheme for the sink nodes and one algorithm that finds the capacity region for the source nodes. The purpose of the first two is to find a difference equation of the following form:

$$P_d(z)\mathbf{y}_d = \mathbf{G}_d(z)\mathbf{u}. \quad (11)$$

This form describes the relationship between the transmitted sequence  $\mathbf{u}[n]$  and the received sequences  $\mathbf{y}_d[n]$  (for every sink node  $d$ ). In (11),  $P_d(z)$  is a polynomial in the time shift operator  $z$ , and  $\mathbf{G}_d(z)$  is a matrix with polynomial elements. These operators are defined in Section II. Using a decoding method similar to the one shown in [7], we can show that it is possible to decode the input sequence from the output when the polynomial  $P_d(z)$  is not the zero polynomial and the *transfer matrix*  $\mathbf{G}_d(z)$  is of full column rank over the polynomial ring  $\mathbb{F}[z]$ .

The difference between the two initialization algorithms is that in the first, it is assumed that we can perform a reset operation on the network at some fixed times and, therefore, this algorithm is a bit faster than the second algorithm that does not operate under this assumption. The purpose of the third algorithm is to find achievable rates for all source nodes. This is done by examining the transfer matrix  $\mathbf{G}_d(z)$  for every sink node  $d$ . To obtain this matrix, one of the initialization algorithms should be used first.

We now present the first algorithm. Its first part consists of  $(\sum_{s \in \mathcal{S}} R_s)$  loops. Every loop takes  $2N + 1$  time units, and after each loop the symbols on all edges are cleared. Algorithm 1 is applied in Example 4 in the appendix.

---

**Algorithm 1** Initialization algorithm with network resetting

---

1) For every  $s \in \mathcal{S}$ , and for every  $j \in \{1, 2, \dots, R_s\}$  do the following:

- Send the sequence  $\mathbf{u}_s^j[n] = (u_{s,1}^j[n], \dots, u_{s,R_s}^j[n])^T$  at the times  $n = 0, 1, \dots, 2N$ ,

where

$$u_{s,i}^j[n] = \begin{cases} 1, & i = j \quad \text{and} \quad n = 0 \\ 0, & i \neq j \quad \text{or} \quad 1 \leq n \leq 2N \end{cases} \quad \forall i \in \{1, \dots, R_s\}. \quad (12)$$

- For all source nodes  $\tilde{s} \neq s$ , send zeros on their input sequences:  $\mathbf{u}_{\tilde{s}}^j[n] = \mathbf{0}$ .
- Every sink node  $d$  should store its received vectors  $\{\mathbf{y}_d^{s,j}[n]\}_{n \in \{0, \dots, 2N\}, s \in \mathcal{S}, j \in \{1, \dots, R_s\}}$ , where each vector  $\mathbf{y}_d^{s,j}[n]$  is of dimension  $l_d$ .
- Reset the network after  $n = 2N$ , by setting  $n = 0$  and  $\mathbf{x}[0] = \mathbf{0}$ .

2) For every sink node  $d$  do the following:

- Combine the received vectors into matrices of size  $l_d \times m$ :

$$\mathbf{M}_d[n] = [\mathbf{y}_d^{s_1,1}[n], \dots, \mathbf{y}_d^{s_1,R_{s_1}}[n], \mathbf{y}_d^{s_2,1}[n], \dots, \mathbf{y}_d^{s_2,R_{s_2}}[n], \dots, \mathbf{y}_d^{s_{|\mathcal{S}|},R_{s_{|\mathcal{S}|}}}[n]]. \quad (13)$$

- Find any non trivial solution to the set of linear equations

$$\sum_{k=0}^N \alpha_{d,k} \mathbf{M}_d[k + \tau] = \mathbf{0}, \forall \tau = 1, \dots, N, \quad (14)$$

where  $\mathbf{0}$  is the  $l_d \times m$  zero matrix and  $\{\alpha_{d,k}\}_{k=0}^N \subseteq \mathbb{F}$  are the unknowns. This set has  $l_d \times m \times N$  equations and it has always a non trivial solution.

---

- 
- Construct the polynomial  $P_d(z)$  and the matrix  $\mathbf{G}_d(z)$  as

$$P_d(z) = \sum_{k=0}^N \alpha_{d,k} z^k, \quad (15)$$

$$\mathbf{G}_d(z) = \sum_{k=1}^N \sum_{j=k}^N \alpha_{d,j} \mathbf{M}_d[j - k + 1] z^{k-1} + \mathbf{M}_d[0] P_d(z). \quad (16)$$

- The difference equation that describes the relationship between the input and the output sequences  $\mathbf{u}[n]$  and  $\mathbf{y}_d[n]$  is given in (11), with the polynomial  $P_d(z)$  and the matrix  $\mathbf{G}_d(z)$  as defined in (15-16). If  $\mathbf{G}_d(z)$  is of full column rank over the polynomial ring  $\mathbb{F}[z]$ , then  $\mathbf{u}[n]$  can be decoded from  $\mathbf{y}_d[n]$  by solving (11). Otherwise, the transmission rates  $\{R_s\}$  of some source nodes should be reduced, or other local encoding kernels should be chosen.
- 

We now present the second algorithm, in which no resetting operation is needed. We consider the case when the network initial state is  $\mathbf{x}_0 \neq \mathbf{0}$  and  $\mathbf{x}_0$  is unknown. Algorithm 2 is similar to the first, except that this algorithm takes additional  $2N + 1$  time units (only in case  $\mathbf{x}_0 \neq \mathbf{0}$ ) and the expression for obtaining  $\mathbf{G}_d(z)$  is a bit different. If  $\mathbf{x}_0 = \mathbf{0}$  then we can skip the operations in the first  $(2N + 1)$  time units since the measured output vectors will contain only zeros. Algorithm 2 is applied in Example 5 in the appendix.

---

**Algorithm 2** Initialization algorithm without network resetting

---

- 1) The input sequence  $\mathbf{u}[n] = (u_1[n], \dots, u_m[n])^T$  that should be sent is

$$u_i[n] = \begin{cases} 1, & n = (2N + 1)i \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in \{1, \dots, m\}, \quad 0 \leq n < (m + 1)(2N + 1), \quad (17)$$

where  $m$  is the dimension of  $\mathbf{u}[n]$ . Note that to send the above sequence, every source node  $s \in \mathcal{S}$  should send the symbol 1 on every one of its inputs in turn  $(u_{s,1}, \dots, u_{s,R_s})$  at the correct time, and zeros at all other times.

---

2) For every sink node  $d$  do the following:

- Find any non trivial solution to the set of linear equations

$$\sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + \tau] = \mathbf{0}, \quad \forall \tau \in \bigcup_{p=0}^m \bigcup_{\tilde{\tau}=1}^N \{(2N+1)p + \tilde{\tau}\}, \quad (18)$$

where  $\{\alpha_{d,j}\}_{j=0}^N \subseteq \mathbb{F}$  are the unknowns. This set has  $l_d \times N \times (m+1)$  equations, and it always has a non trivial solution.

- The polynomial  $P_d(z)$  and the matrix  $\mathbf{G}_d(z)$  are defined below:

$$P_d(z) = \sum_{k=0}^N \alpha_{d,k} z^k, \quad (19)$$

$$\mathbf{g}_{d,i}(z) = \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + (2N+1)i - k + 1] z^{k-1}, \quad \forall i \in \{1, \dots, m\}, \quad (20)$$

$$\mathbf{G}_d(z) = [\mathbf{g}_{d,1}(z), \mathbf{g}_{d,2}(z), \dots, \mathbf{g}_{d,m}(z)]. \quad (21)$$

- The difference equation that describes the relationship between the input and the output sequences  $\mathbf{u}[n]$  and  $\mathbf{y}_d[n]$  for  $n \geq 1$  is given in (11), with the polynomial  $P_d(z)$  and the matrix  $\mathbf{G}_d(z)$  as defined in (19-21). If  $\mathbf{G}_d(z)$  is of full column rank over the polynomial ring  $\mathbb{F}[z]$ , then  $\mathbf{u}[n]$  can be decoded from  $\mathbf{y}_d[n]$  by solving (11). Otherwise, the transmission rates  $\{R_s\}$  of some source nodes should be reduced, or other local encoding kernels should be chosen.

We now present the third algorithm that allows us to find achievable rates for all source nodes in the network, with the chosen local encoding kernels. It uses the matrix  $\mathbf{G}_d(z)$  from (11) and hence Algorithm 1 or 2 should be used first to find the matrix. At the end of this algorithm, every sink node  $d$  will be able to tell what rates are achievable for it.

Algorithm 3 allows us to find achievable rates with the currently chosen encoding kernels. If they were chosen randomly from a large enough field, these rates will be, with high probability, all the rates from the capacity region. There is, however, a small

---

**Algorithm 3** Finding the capacity region
 

---

The capacity region is found as follows:

- For every sink node  $d$ , split the matrix  $\mathbf{G}_d(z)$  into  $|\mathcal{S}|$  matrices, such that each matrix  $\mathbf{G}_{d,s}(z)$  has  $R_s$  columns and such that the following will hold:

$$\begin{aligned} \mathbf{G}_d(z)\mathbf{u} &= \left[ \mathbf{G}_{d,s_1}(z), \dots, \mathbf{G}_{d,s_{|\mathcal{S}|}}(z) \right] \begin{bmatrix} \mathbf{u}_{s_1} \\ \vdots \\ \mathbf{u}_{s_{|\mathcal{S}|}} \end{bmatrix} \\ &= \sum_{s \in \mathcal{S}} \mathbf{G}_{d,s}(z) \mathbf{u}_s. \end{aligned} \quad (22)$$

- For every possible  $n$ -tuple  $(R'_s)_{s \in \mathcal{S}}$  with integer entries that satisfy  $R'_s \leq R_s$ , check if for every source node  $s$ , there exist  $R'_s$  column vectors  $\{\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}\}$  in the columns of the matrix  $\mathbf{G}_{d,s}(z)$  such that all the vectors  $\cup_{s \in \mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$  are linearly independent over the polynomial ring  $\mathbb{F}[z]$ . If there are such vectors, the rates  $(R'_s)_{s \in \mathcal{S}}$  are achievable for the sink node  $d$ .
  - The capacity region is obtained by taking all  $n$ -tuples  $(R'_s)_{s \in \mathcal{S}}$  that are achievable for every sink node.
- 

probability that the local encoding kernels were not chosen well. In that case, Algorithm 3 will only give the achievable rates with the currently chosen coefficients. Algorithm 3 is applied in Example 6 in the appendix.

**Remark 1** Although we restricted ourselves to the case of scalar local encoding kernels, the algorithms can be extended to networks that use CNC with rational power series as local encoding kernels [9, p. 492]. In this case the input-output relationship of each node  $j \in \mathcal{V}$  can be described by state space equations [14, p. 481]. Denote the state vector of node  $j$  by  $\tilde{\mathbf{x}}_j[n]$ , and its dimension by  $\dim \tilde{\mathbf{x}}_j[n]$ . If we concatenate all state vectors  $\{\tilde{\mathbf{x}}_j[n]\}_{j \in \mathcal{V}}$  into one state vector  $\tilde{\mathbf{x}}[n]$  of dimension  $\sum_{j \in \mathcal{V}} \dim(\tilde{\mathbf{x}}_j[n])$ , a global state space representation of the network can be written:

$$\tilde{\mathbf{x}}[n+1] = \hat{\mathbf{A}}\tilde{\mathbf{x}}[n] + \hat{\mathbf{B}}\mathbf{u}[n], \quad (23)$$

$$\mathbf{y}_d[n] = \hat{\mathbf{C}}_d \tilde{\mathbf{x}}[n] + \mathbf{D}_d \mathbf{u}[n], \quad (24)$$

where  $\hat{\mathbf{A}}$ ,  $\hat{\mathbf{B}}$ ,  $\hat{\mathbf{C}}_d$  and  $\mathbf{D}_d$  are defined by the network topology and the local encoding kernels. The derivation of our algorithms is based only on the fact that the input-output relationship of the network can be written as state space equations with a state vector of dimension  $|\mathcal{E}| \leq N$ . In the case where we use rational power series as local encoding kernels, the algorithms will still apply if we take  $N$  to be larger than the dimension of the state vector  $\tilde{\mathbf{x}}$ :

$$N \geq \sum_{j \in \mathcal{V}} \dim(\tilde{\mathbf{x}}_j[n]). \quad (25)$$

#### IV. DERIVATION OF ALGORITHM 1

Our goal is to find a relationship between the input sequence  $\mathbf{u}[n]$  and the output sequence  $\mathbf{y}_d[n]$  for every sink node  $d$  that will allow it to decode the sent symbols. Such a relationship can be given in the form of a difference equation, similar to that given in (11). The problem is how to find a polynomial  $P_d(z)$  and a matrix  $\mathbf{G}_d(z)$  that will satisfy (11) for all  $n \geq 0$  only from the received symbols  $\mathbf{y}_d[n]$ . We assume, without loss of generality, that  $N$  is equal to the number of edges in the network. However, if  $N$  is larger, we can assume that there are an additional  $2(N - |\mathcal{E}|)$  virtual nodes and  $(N - |\mathcal{E}|)$  virtual edges between these nodes. The virtual edges are not connected to the original network and have no influence on it. In this way, the number of edges in the new network is  $N$ . We observe that in view of (5) and by the definition of  $\mathbf{x}[n]$ ,  $\mathbf{u}[n]$  and  $\mathbf{y}_d[n]$ , for every sink node  $d$ , a state space representation of the network can be written as

$$\mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n], \quad \mathbf{x}[0] = \mathbf{x}_0, \quad \forall n \geq 0, \quad (26)$$

$$\mathbf{y}_d[n] = \mathbf{C}_d \mathbf{x}[n] + \mathbf{D}_d \mathbf{u}[n], \quad \forall n \in \mathbb{Z}, \quad (27)$$

where the matrices  $\mathbf{A}$  and  $\mathbf{B}$  are of sizes  $N \times N$  and  $N \times m$ , respectively, and are determined by the network structure and the local encoding kernels on every node. An example of the matrices  $\mathbf{A}$  and  $\mathbf{B}$  is shown in Example 1. The matrices  $\mathbf{C}_d$  and  $\mathbf{D}_d$  contain

only ones and zeros and are chosen so that  $\mathbf{y}_d[n]$  will contain the incoming symbols and the symbols generated by  $d$ , if  $d$  is a source node.

A general solution to the state equations (26) and (27) is given by

$$\mathbf{y}_d[n] = \mathbf{C}_d \mathbf{A}^n \mathbf{x}_0 + \sum_{i=0}^{n-1} \mathbf{C}_d \mathbf{A}^{n-1-i} \mathbf{B} \mathbf{u}[i] + \mathbf{D}_d \mathbf{u}[n], \quad \forall n \geq 0. \quad (28)$$

In Algorithm 1, it is assumed that  $\mathbf{x}_0 = \mathbf{0}$ . After the source nodes send basis vectors, as described in step 1 in the algorithm, every sink node has the matrices given in the following lemma.

**Lemma 1** For a network described by the state equations (26)-(27) with  $\mathbf{x}_0 = \mathbf{0}$ , let the input sequence  $\mathbf{u}[n]$  be given by

$$\mathbf{u}_i[n] = \begin{cases} \mathbf{e}_i, & n = 0 \\ \mathbf{0}, & 1 \leq n \leq 2N \end{cases}, \quad (29)$$

where  $\mathbf{e}_i$  is a basis vector of the form

$$\mathbf{e}_i = (a_0, a_1, \dots, a_m)^T, \quad a_k = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases}. \quad (30)$$

The output sequence in that case will be

$$\mathbf{y}_{d,i}[n] = \begin{cases} \mathbf{D}_d \cdot \mathbf{e}_i, & n = 0 \\ \mathbf{C}_d \mathbf{A}^{n-1} \mathbf{B} \cdot \mathbf{e}_i, & 1 \leq n \leq 2N \end{cases}. \quad (31)$$

Moreover, if one combines the output vectors into matrices  $\mathbf{M}_d[n] = [\mathbf{y}_{d,1}[n], \dots, \mathbf{y}_{d,m}[n]]$ , then the corresponding matrices will be

$$\begin{aligned} \mathbf{M}_d[n] &= \mathbf{C}_d \mathbf{A}^{n-1} \mathbf{B}, \quad \forall 1 \leq n \leq 2N, \\ \mathbf{M}_d[0] &= \mathbf{D}_d. \end{aligned} \quad (32)$$

*Proof:* The proof for this lemma follows immediately by substituting the input sequence from (29) into the general solution given in (28) and using the fact that the initial state  $\mathbf{x}_0$  is zero. ■



The above matrices  $\{\mathbf{M}_d[n]\}$  are usually called the *Markov parameters* of an LTI system. To continue, we state the Cayley-Hamilton [17, p. 284] Theorem, since it plays an important role in our derivation.

**Theorem 1 (Cayley-Hamilton Theorem)** For a given  $n \times n$  matrix  $\mathbf{A}$  over the field  $\mathbb{F}$ , let  $P_{\mathbf{A}}(t) = \det(t\mathbf{I} - \mathbf{A})$  be the characteristic polynomial of  $\mathbf{A}$ . Let  $\{a_k\}_{k=0}^{n-1}$  be the coefficients of  $P_{\mathbf{A}}(t)$ , so that it can be represented as

$$P_{\mathbf{A}}(t) = t^n + \sum_{k=0}^{n-1} a_k t^k. \quad (33)$$

Then the following holds:

$$P_{\mathbf{A}}(\mathbf{A}) = \mathbf{A}^n + \sum_{k=0}^{n-1} a_k \mathbf{A}^k = \mathbf{O}, \quad (34)$$

where  $\mathbf{O}$  is the zero  $n \times n$  matrix.

We now look for a non zero polynomial  $P_d(t) = \sum_{k=0}^N \alpha_{d,k} t^k$  that will satisfy

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau} \mathbf{B} = \mathbf{O}, \quad \forall \tau \in \mathbb{N}. \quad (35)$$

We will show later that this polynomial is used in the difference equation (11), which is needed for decoding the transmitted symbols. The set of linear equations given in (35) has an infinite number of equations; using the Cayley-Hamilton Theorem it has at least one solution, where  $P_d(t)$  is the characteristic polynomial of  $\mathbf{A}$ . It is interesting to note that to find  $P_d(t)$ , we do not need all of these equations since they are linearly dependent. In fact, we have the following lemma that tells us how many equations we need.

**Lemma 2** If a polynomial  $P_d(t)$  satisfies

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau} \mathbf{B} = \mathbf{O}, \quad \forall \tau \in \{0, 1, \dots, N-1\}, \quad (36)$$

where  $\mathbf{A}$  is a square  $N \times N$  matrix, then it also satisfies (35).

The proofs for this lemma and for those of all the other theorems are given in Appendix B. If we denote the unknown polynomial by  $P_d(t) = \sum_{k=0}^N \alpha_{d,k} z^k$ , then the set of linear equations given in (36) can be written as

$$\forall \tau \in \{1, \dots, N\} : \mathbf{O} = \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1} \mathbf{B} \quad (37)$$

$$= \sum_{k=0}^N \alpha_{d,k} \mathbf{C}_d \mathbf{A}^{k+\tau-1} \mathbf{B} \quad (38)$$

$$\stackrel{(a)}{=} \sum_{k=0}^N \alpha_{d,k} \mathbf{M}_d[k + \tau], \quad (39)$$

where (a) is obtained from Lemma 1. We therefore see that the set of linear equations solved in (14) is the same set as in (36). The next theorem describes the relationship between the sent and received symbols in the network.

**Theorem 2** For a given network and a sink node  $d$ , let  $P_d(z) = \sum_{k=0}^N \alpha_{d,k} z^k$  and  $\mathbf{G}_d(z)$  be the polynomial and the matrix defined in (15)-(16). Then (11) holds. Furthermore, it is possible to decode  $\mathbf{u}$  from  $\mathbf{y}_d$  if and only if the matrix  $\mathbf{G}_d(z)$  is of full column rank over the polynomial ring  $\mathbb{F}[z]$ .

Theorem 2 gives us a way to decode the sent symbols, and it assures us that if the set of linear equations in (11) does not have a unique solution, then there is no way for us to find  $\mathbf{u}$  from  $\mathbf{y}_d$ , even if we know the network topology and the local encoding kernels.

## V. DERIVATION OF ALGORITHM 2

We are interested, again, in a difference equation between  $\mathbf{u}$  and  $\mathbf{y}_d$ , as given in (11), that does not depend on  $\mathbf{x}_0$ . Let  $\{\mathbf{e}_k\}_{k=1}^m$  be the standard basis for the vector space  $\mathbb{F}^m$ , namely, the elements of the vector  $\mathbf{e}_k$  are zeros except for the  $k$ 'th element which is equal to one. As described in step 1 of Algorithm 2, the input sequence  $\mathbf{u}[n]$  is given by

$$\mathbf{u}[n] = \sum_{k=1}^m \mathbf{e}_k 1_{\{n=(2N+1)k\}}, \quad (40)$$

where  $1_{\{\cdot\}}$  is the indicator function

$$1_{\Omega} = \begin{cases} 1, & \text{statement } \Omega \text{ is true} \\ 0, & \text{otherwise} \end{cases}.$$

We can get the output sequence if we substitute the above input sequence into the general solution (28) of the network's state equations. The output sequence  $\mathbf{y}_d[n]$  for  $n \geq 0$  will be

$$\mathbf{y}_d[n] = \mathbf{C}_d \mathbf{A}^n \mathbf{x}_0 + \sum_{k=1}^m \sum_{i=0}^{n-1} \mathbf{C}_d \mathbf{A}^{n-1-i} \mathbf{B} \mathbf{e}_k 1_{\{i=(2N+1)k\}} + \mathbf{D}_d \sum_{k=1}^m \mathbf{e}_k 1_{\{n=(2N+1)k\}} \quad (41)$$

$$= \mathbf{C}_d \mathbf{A}^n \mathbf{x}_0 + \sum_{k=1}^{\min\{m, \lfloor \frac{n-1}{2N+1} \rfloor\}} \mathbf{C}_d \mathbf{A}^{n-1-(2N+1)k} \mathbf{B} \mathbf{e}_k + \mathbf{D}_d \mathbf{e}_{n/(2N+1)} 1_{\{n/(2N+1) \in \mathbb{N}\}}. \quad (42)$$

We look for a non zero polynomial  $P_d(t)$  that will satisfy

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^\tau \mathbf{B} = \mathbf{O}, \quad \forall \tau \geq 0, \quad (43)$$

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau+1} \mathbf{x}_0 = \mathbf{0}, \quad \forall \tau \geq 0, \quad (44)$$

where  $\mathbf{O}$  is the  $l_d \times m$  zero matrix and  $\mathbf{0}$  is the zero column vector of dimension  $l_d$ . This polynomial is used in the difference equation (11), which is needed for decoding the transmitted symbols. We can limit ourselves to  $\tau \in \{0, \dots, N-1\}$ , as can be seen by the next lemma.

**Lemma 3** Let  $P_d(t)$  be a polynomial in  $t$  and  $\mathbf{A}$  is a square  $N \times N$  matrix. If either of the equations (43) or (44) hold for  $\tau \in \{0, \dots, N-1\}$ , then it also holds for all  $\tau \geq N$ .

The proof is similar to the proof for Lemma 2 and is therefore omitted. A method for finding such a polynomial from the received sequence  $\{\mathbf{y}_d[n]\}_{1 \leq n < (m+1)(2N+1)}$  is given in the next theorem.

**Theorem 3** The polynomial  $P_d(t) = \sum_{k=0}^N \alpha_{d,k} t^k$  satisfies (43)-(44) if and only if its coefficients are a solution of (18).

Using the Cayley-Hamilton Theorem, there is at least one polynomial that satisfies (43)-(44), which is the characteristic polynomial of  $\mathbf{A}$ , so (18) has at least one solution. After finding a polynomial  $P_d(t)$ , we can construct a difference equation for  $\mathbf{u}$  and  $\mathbf{y}_d$  that does not depend on the initial state  $\mathbf{x}_0$ . The equation will hold for any time after the initialization process finishes, even without resetting the state vector.

**Theorem 4** Let  $P_d(z)$  and  $\mathbf{G}_d(z)$  be the polynomial and the matrix defined in (19)-(21). Then the following difference equation holds:

$$(P_d(z) \mathbf{y}_d)[n] = (\mathbf{G}_d(z) \mathbf{u})[n], \quad \forall n \geq 1. \quad (45)$$

Equation (45) itself is not enough for a decoding algorithm since it holds only for  $n \geq 1$ . In order to decode we need the first  $N$  values of  $\mathbf{u}$ :  $\mathbf{u}[0], \dots, \mathbf{u}[N]$  to be known a

priori to the sink nodes. Note that if we apply Algorithm 2 these values are zeros and, hence, are known a priori. We define:

$$\mathbf{q}[n] = \begin{cases} (\mathbf{G}_d(z)\mathbf{u})[n], & n \leq 0 \\ P_d(z)\mathbf{y}_d(z), & n \geq 1 \end{cases}. \quad (46)$$

Note that  $(\mathbf{G}_d(z)\mathbf{u})[n]$  can be calculated for  $n \leq 0$  if we know  $\mathbf{u}[0], \dots, \mathbf{u}[N]$ , since

$$(\mathbf{G}_d(z)\mathbf{u})[n] = \sum_{k=0}^N \mathbf{G}_d[k]\mathbf{u}[n+k]. \quad (47)$$

Once we have the sequence  $\mathbf{q}$ , we note that it satisfies

$$\mathbf{q}[n] = (\mathbf{G}_d(z)\mathbf{u})[n], \quad \forall n \in \mathbb{Z}, \quad (48)$$

so we can use (99-100) to find  $\mathbf{u}[n]$  (if  $\mathbf{G}_d(z)$  is of full column rank).

## VI. DERIVATION OF ALGORITHM 3

A direct consequence of Theorem 2 is the fact that we can find achievable rates for every source node from the matrices  $\{\mathbf{G}_d(z)\}_{d \in \mathcal{D}}$ . If the transmission rates are not achievable with the given local encoding kernels, then one cannot decode the input sequence  $\mathbf{u}$  from the output  $\mathbf{y}_d$ . This result is stated in the following theorem.

**Theorem 5** For a given network and a sink node  $d$ , let

$$P_d(z)\mathbf{y}_d = \mathbf{G}_d(z)\mathbf{u} = \sum_{s \in \mathcal{S}} \mathbf{G}_{d,s}(z)\mathbf{u}_s. \quad (49)$$

describe the relationship between the input sequence  $\mathbf{u}$  and the output sequence  $\mathbf{y}_d$  that was found in Algorithm 1 or 2. For every source node  $s \in \mathcal{S}$ , let  $R_s$  be the transmission rate of  $s$  that was set before the initialization algorithm was started. Then the rates  $(R'_s)_{s \in \mathcal{S}}$  are achievable for the sink node  $d$  with the current local encoding kernels if and only if for every source node  $s \in \mathcal{S}$  there exist  $R'_s$  linearly independent column vectors  $\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}$  from the columns of the matrix  $\mathbf{G}_{d,s}(z)$ , such that  $\cup_{s \in \mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$  is a set of linearly independent vectors over the polynomial ring  $\mathbb{F}[z]$ .

## VII. CONCLUSIONS

The use of CNC schemes requires one to choose local encoding kernels at the relay nodes that would allow the sink nodes to decode the transmitted symbols. The coefficients can be chosen randomly to simplify the network code construction, but this would require the sink nodes to know the transfer function of the network. The algorithms we presented allow the sink nodes to find a difference equation that enables decoding the transmitted from the received symbols without learning the exact topology of the network and the chosen local encoding kernels. The capacity region can also be found from the obtained difference equation. The algorithms require the source nodes to transmit basis vectors and the sink nodes to solve a set of linear equations. Both the amount of transmissions every source node needs to perform and the number of linear equations every sink node needs to solve grow linearly with the number of edges and hence, the algorithms are considered efficient computationally.

## APPENDIX A

### EXAMPLES

**Example 4** Consider the network shown in Fig.3, with two source nodes  $s_1, s_2$ , one sink node  $d$  and three relay nodes. The field on which the network operates is  $\mathbb{F}_{2^8}$  with the primitive polynomial  $t^8 + t^4 + t^3 + t^2 + 1$  used to define the field. The elements of the field  $\mathbb{F}_{2^8}$  are polynomials of the form:

$$\sum_{k=0}^7 a_k t^k, \quad \forall i : a_i \in \{0, 1\}. \quad (50)$$

For simplicity, we use an integer representation for every scalar from the field, such that every scalar is represented by a number between 0 and 255 whose binary representation  $(a_7, a_6, \dots, a_0)$  is given by the elements  $a_i$  from (50).

All local encoding kernels were generated randomly and are given by

$$x_1[n+1] = 37x_6[n] + 108x_3[n], \quad (51)$$

$$x_2[n+1] = 234x_1[n] + 203x_8[n], \quad (52)$$

$$x_3[n+1] = 245x_7[n] + 168x_2[n], \quad (53)$$

$$x_4[n+1] = 10x_1[n] + 217x_8[n], \quad (54)$$

$$x_5[n+1] = 239x_7[n] + 174x_2[n], \quad (55)$$

$$x_6[n+1] = 194u_{s_1,1}[n] + 190u_{s_1,2}[n], \quad (56)$$

$$x_7[n+1] = 101u_{s_1,1}[n] + 168u_{s_1,2}[n], \quad (57)$$

$$x_8[n+1] = 44u_{s_2,1}[n]. \quad (58)$$

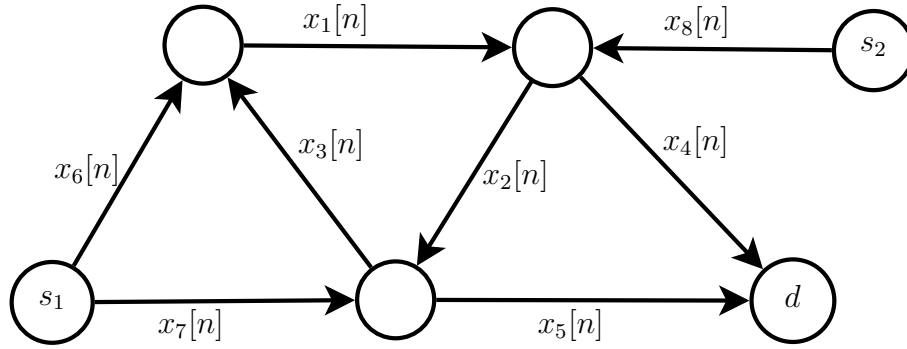


Fig. 3. Network with 2 source nodes, one sink node and 3 relay nodes.

All nodes know only the following facts:

- The source nodes list is  $\mathcal{S} = \{s_1, s_2\}$  and the sink node is  $d$ .
- The network has not more than 8 edges ( $N = 8$ ).
- The number of output links for every source node:  $|Out(s_1)| = 2$  and  $|Out(s_2)| = 1$ .

Note that even though the rates  $(R_{s_1}, R_{s_2}) = (2, 1)$  are not achievable (they do not satisfy the Min-Cut Max-Flow condition), we assume for now that this information is not known a priori. If it was, we could set the rates to  $(R_{s_1}, R_{s_2}) = (1, 1)$  (by setting  $u_{s_1,2}[n] = 0$ ) or to  $(R_{s_1}, R_{s_2}) = (2, 0)$  (by setting  $u_{s_2,1}[n] = 0$ ), since these rates are achievable. In that case the whole initialization process would take 34 time units.. The initialization process begins when  $s_1$  and  $s_2$  send the following sequences:

$$u_{s_1,1}[n] = 1, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16, \quad (59)$$

$$u_{s_1,2}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16,$$

$$u_{s_2,1}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16.$$

After  $n = 16$ , all the incoming symbols are cleared,  $n$  is set to zero, and the following sequences are sent:

$$u_{s_1,1}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16, \quad (60)$$

$$u_{s_1,2}[n] = 1, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16,$$

$$u_{s_2,1}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16.$$

Again, after  $n = 16$  the network is cleared, and the sent sequences are:

$$u_{s_1,1}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16, \quad (61)$$

$$u_{s_1,2}[n] = 0, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16,$$

$$u_{s_2,1}[n] = 1, 0, 0, \dots, 0 \quad \forall 0 \leq n \leq 16.$$

Meanwhile, the output sequence  $\mathbf{y}_d = (x_4, x_5)^T$  received by  $d$  is given by

$$\begin{aligned} \mathbf{M}_d[n] &= [\mathbf{y}_d^{s_1,1}[n], \mathbf{y}_d^{s_1,2}[n], \mathbf{y}_d^{s_2,1}[n]] \quad \forall n \in \{1, \dots, 16\} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 231 \\ 157 & 13 & 0 \end{bmatrix}, \begin{bmatrix} 57 & 73 & 0 \\ 0 & 0 & 228 \end{bmatrix}, \begin{bmatrix} 113 & 63 & 0 \\ 185 & 105 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 228 \\ 1 & 101 & 0 \end{bmatrix}, \dots \end{aligned} \quad (62)$$

and we have

$$\mathbf{M}_d[n+3] = 209\mathbf{M}_d[n] \quad \forall 3 \leq n \leq 13. \quad (63)$$

We now solve the set of linear equations given in (14). We look for some non trivial solution. We can take, for example,

$$P(z) = \sum_{k=0}^8 \alpha_{d,k} z^k \quad (64)$$

$$= 209z^2 + z^5. \quad (65)$$

This is, indeed, a solution of (14), as can be seen from (63). The matrix  $\mathbf{G}_d(z)$  given by this solution is:

$$(209\mathbf{M}_d[2] + \mathbf{M}_d[5]) + (209\mathbf{M}_d[1] + \mathbf{M}_d[4])z + (\mathbf{M}_d[3])z^2 + (\mathbf{M}_d[2])z^3 + (\mathbf{M}_d[1])z^4 =$$

$$= \begin{bmatrix} 113z + 57z^2 & 63z + 73z^2 & 84 + 231z^3 \\ 24 + 185z + 157z^3 & 17 + 105z + 13z^3 & 228z^2 \end{bmatrix}. \quad (66)$$

If we are interested in finding achievable rates from the matrix  $\mathbf{G}_d(z)$ , we should apply Algorithm 3, as described in Example 6. For now, we set the rates to achievable ones:  $R_{s_1} = R_{s_2} = 1$  (by sending  $u_{s_1,2}[n] = 0$ ), and we get the following relationship between the input and the output sequences:

$$(z^5 + 209 \cdot z^2)\mathbf{y}_d = \begin{bmatrix} 113z + 57z^2 & 84 + 231z^3 \\ 24 + 185z + 157z^3 & 228z^2 \end{bmatrix} \begin{pmatrix} u_{s_1,1} \\ u_{s_2,1} \end{pmatrix}. \quad (67)$$

Note that if we had started the initialization process with the rates  $R_{s_1} = R_{s_2} = 1$ , we would have obtained the following transfer matrix  $\tilde{\mathbf{G}}_d(z)$ :

$$\tilde{\mathbf{G}}_d(z) = \begin{bmatrix} 113z + 57z^2 & 84 + 231z^3 \\ 24 + 185z + 157z^3 & 228z^2 \end{bmatrix}. \quad (68)$$

We can solve (67) for  $\mathbf{u}$  by multiplying both sides of the equation by  $(42 \cdot \text{adj}(\tilde{\mathbf{G}}_d(z)))$ :

$$\begin{aligned} (105 + 223z + 152z^3 + 149z^4 + z^6) \begin{pmatrix} u_{s_1,1} \\ u_{s_2,1} \end{pmatrix} &= \\ &= \begin{bmatrix} 221z^4 + 119z^7 & 65z^2 + 119z^5 + 9z^8 \\ 30z^2 + 208z^3 + 42z^5 + 112z^6 + 241z^8 & 42z^3 + 112z^4 + 203z^6 + 212z^7 \end{bmatrix} \mathbf{y}_d, \end{aligned} \quad (69)$$

where we used the identity

$$42 \cdot \text{adj}(\tilde{\mathbf{G}}_d(z)) \tilde{\mathbf{G}}_d(z) = 42 \det(\tilde{\mathbf{G}}_d(z)) \mathbf{I}. \quad (70)$$

We used the factor 42 to make the coefficient of  $z^6$  from the left side of (69) equal one.

The difference equation for  $\mathbf{u}[n]$  is:

$$\begin{aligned} \begin{pmatrix} u_{s_1,1}[n+6] \\ u_{s_2,1}[n+6] \end{pmatrix} &= 149 \begin{pmatrix} u_{s_1,1}[n+4] \\ u_{s_2,1}[n+4] \end{pmatrix} + 152 \begin{pmatrix} u_{s_1,1}[n+3] \\ u_{s_2,1}[n+3] \end{pmatrix} + 223 \begin{pmatrix} u_{s_1,1}[n+1] \\ u_{s_2,1}[n+1] \end{pmatrix} + 105 \begin{pmatrix} u_{s_1,1}[n] \\ u_{s_2,1}[n] \end{pmatrix} \\ &+ \begin{pmatrix} 221y_{d,1}[n+4] + 119y_{d,1}[n+7] \\ 30y_{d,1}[n+2] + 208y_{d,1}[n+3] + 42y_{d,1}[n+5] + 112y_{d,1}[n+6] + 241y_{d,1}[n+8] \end{pmatrix} \end{aligned}$$



$$+ \begin{pmatrix} 65y_{d,2}[n+2] + 119y_{d,2}[n+5] + 9y_{d,2}[n+8] \\ 42y_{d,2}[n+3] + 112y_{d,2}[n+4] + 203y_{d,2}[n+6] + 212y_{d,2}[n+7] \end{pmatrix}, \quad (71)$$

with the initial conditions

$$\mathbf{u}[n] = \mathbf{0}, \forall n < 0, \quad (72)$$

$$\mathbf{y}_d[n] = \mathbf{0}, \forall n < 0. \quad (73)$$

**Example 5** We again look at the network in Fig.3, with the field  $\mathbb{F}_{2^8}$  and the same local encoding kernels as in the previous example. Now we assume there is an initial non zero state for the network:

$$(x_1[0], \dots, x_8[0])^T = (50, 64, 157, 121, 90, 212, 149, 140)^T. \quad (74)$$

We follow the instructions of Algorithm 2 to get a difference equation for  $\mathbf{u}$  and  $\mathbf{y}_d$ . As in Example 4, we assume that achievable rates are not known yet and, therefore, we set the transmission rates to  $R_{s_1} = 2$  and  $R_{s_2} = 1$ . At first, the source nodes  $s_1$  and  $s_2$  transmit the following sequences:

$$\begin{aligned} u_{s_1,1}[n] &= \begin{cases} 1, & n = 17 \\ 0, & \text{otherwise} \end{cases} \quad \forall 0 \leq n < 68, \\ u_{s_1,2}[n] &= \begin{cases} 1, & n = 34 \\ 0, & \text{otherwise} \end{cases} \quad \forall 0 \leq n < 68, \\ u_{s_2,1}[n] &= \begin{cases} 1, & n = 51 \\ 0, & \text{otherwise} \end{cases} \quad \forall 0 \leq n < 68. \end{aligned} \quad (75)$$

The output sequence  $\{\mathbf{y}_d[n]\}_{1 \leq n < 68}$  is stored at the sink node  $d$ . Here are some of the initial and final values of  $\{\mathbf{y}_d[n]\}$ :

$$\mathbf{y}_d[n] = \begin{pmatrix} 164 \\ 96 \end{pmatrix}, \begin{pmatrix} 253 \\ 6 \end{pmatrix}, \begin{pmatrix} 155 \\ 88 \end{pmatrix}, \dots, \begin{pmatrix} 97 \\ 254 \end{pmatrix} \quad \forall 1 \leq n \leq 19, \quad (76)$$

$$\begin{pmatrix} 63 \\ 144 \end{pmatrix}, \begin{pmatrix} 18 \\ 101 \end{pmatrix}, \begin{pmatrix} 144 \\ 46 \end{pmatrix}, \dots, \begin{pmatrix} 225 \\ 209 \end{pmatrix}, \begin{pmatrix} 172 \\ 108 \end{pmatrix}, \quad \forall 20 \leq n \leq 67. \quad (77)$$

A solution of (18) leads to the same solution as in the previous example and, therefore the same decoding method can be used.

$$\begin{aligned} P_d(z) &= \sum_{j=0}^N \alpha_{d,j} z^j \\ &= z^5 + 209z^2, \\ \mathbf{G}_d(z) &= \begin{bmatrix} 113z + 57z^2 & 63z + 73z^2 & 84 + 231z^3 \\ 24 + 185z + 157z^3 & 17 + 105z + 13z^3 & 228z^2 \end{bmatrix}. \end{aligned}$$

**Example 6** We return to the network in Fig. 3, with the same field and coefficients as in Example 4. After applying Algorithm 1 or 2, we get the polynomial  $P_d(z)$  and the transfer matrix  $\mathbf{G}_d(z)$ , as given in (65) and (66). We are interested in achievable rates for the sources  $s_1, s_2$ , so we follow the instructions given in Algorithm 3. We split  $\mathbf{G}_d(z)$  into two matrices:

$$\mathbf{G}_{d,s_1}(z) = \begin{bmatrix} 113z + 57z^2 & 63z + 73z^2 \\ 24 + 185z + 157z^3 & 17 + 105z + 13z^3 \end{bmatrix}, \quad \mathbf{G}_{d,s_2}(z) = \begin{bmatrix} 84 + 231z^3 \\ 228z^2 \end{bmatrix}. \quad (78)$$

The rates  $(R_{s_1}, R_{s_2}) = (1, 1)$  are achievable, since the vectors

$$\mathbf{v}_1 = (113z + 57z^2, 24 + 185z + 157z^3)^T \in \text{Columns of } (\mathbf{G}_{d,s_1}), \quad (79)$$

$$\mathbf{v}_2 = (84 + 231z^3, 228z^2)^T \in \text{Columns of } (\mathbf{G}_{d,s_2}) \quad (80)$$

are linearly independent over the polynomial ring  $\mathbb{F}[z]$ . The rates  $(R_{s_1}, R_{s_2}) = (2, 0)$  are also achievable, since  $\mathbf{G}_{d,s_1}$  is of full rank over the polynomial ring  $\mathbb{F}[z]$ .

## APPENDIX B

### PROOFS

*Proof for Lemma 2:* A direct consequence of the Cayley-Hamilton Theorem is that for every  $N \times N$  matrix  $\mathbf{A}$ , its power  $\mathbf{A}^T$  can be written as a linear combination of

$\mathbf{I}, \mathbf{A}, \mathbf{A}^2, \dots, \mathbf{A}^{N-1}$  for  $\tau \geq N$ . Therefore, by substituting this into (35) we get for every  $\tau \geq N$ :

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^\tau \mathbf{B} = \mathbf{C}_d P_d(\mathbf{A}) \sum_{i=0}^{N-1} \gamma_i \mathbf{A}^i \mathbf{B} \quad (81)$$

$$= \sum_{i=0}^{N-1} \gamma_i (\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^i \mathbf{B}) \quad (82)$$

$$= \mathbf{O}, \quad (83)$$

where the last equality holds because  $P_d(z)$  satisfies (36). ■

*Proof for Theorem 2:* We note first that if the network is described by the state-space equations (26)-(27), then, by Lemmas 1-2, the polynomial  $P_d(z) = \sum_{k=0}^N \alpha_{d,k} z^k$  found in Algorithm 1 satisfies (35), and the transfer matrix  $\mathbf{G}_d(z)$  is given by

$$\mathbf{G}_d(z) = \sum_{k=1}^N \sum_{j=k}^N \alpha_{d,j} \mathbf{M}_d[j-k+1] z^{k-1} + \mathbf{M}_d[0] P_d(z) \quad (84)$$

$$= \sum_{k=1}^N \left( \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \right) z^{k-1} + P_d(z) \mathbf{D}_d, \quad (85)$$

where  $\{\mathbf{M}_d[n]\}$  are the Markov parameters of the network. From (28), for every  $n \geq 0$  we have:

$$\mathbf{y}_d[n+1] = \sum_{i=0}^n \mathbf{C}_d \mathbf{A}^{n-i} \mathbf{B} \mathbf{u}[i] + \mathbf{D}_d \mathbf{u}[n+1]. \quad (86)$$

By applying the  $P_d(z)$  operator on both sides we get:

$$(P_d(z) \mathbf{y}_d - P_d(z) \mathbf{D}_d \mathbf{u})[n+1] = \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[n+j+1] - (P_d(z) \mathbf{D}_d \mathbf{u})[n+1]. \quad (87)$$

By expanding  $\mathbf{y}_d[n+j+1]$  and changing the summation order, we get:

$$(P_d(z) \mathbf{y}_d - P_d(z) \mathbf{D}_d \mathbf{u})[n+1] = \quad (88)$$

$$= \sum_{j=0}^N \sum_{i=0}^{n+j} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{n+j-i} \mathbf{B} \mathbf{u}[i] \quad (89)$$

$$\stackrel{(a)}{=} \left( \sum_{i=0}^n \sum_{j=0}^N + \sum_{i=n+1}^{n+N} \sum_{j=i-n}^N \right) \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{n+j-i} \mathbf{B} \mathbf{u}[i] \quad (90)$$

$$= \sum_{i=0}^n \mathbf{C}_d \left( \sum_{j=0}^N \alpha_{d,j} \mathbf{A}^j \right) \mathbf{A}^{n-i} \mathbf{B} \mathbf{u}[i] + \sum_{i=n+1}^{n+N} \left( \sum_{j=i-n}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{n+j-i} \mathbf{B} \right) \mathbf{u}[i] \quad (91)$$

$$\stackrel{(b)}{=} 0 + \sum_{k=1}^N \left( \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \right) \mathbf{u}[n+k] \quad (92)$$

$$= \left( \sum_{k=1}^N \left( \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \right) z^k \mathbf{u} \right) [n], \quad (93)$$

where

(a) is obtained by changing the summation order,

(b) follows from the fact that  $P_d(z)$  satisfies (35) and by changing a summation variable

$$k = i - n.$$

Therefore, we get:

$$P_d(z) \mathbf{y}_d = \left( \sum_{k=1}^N \left( \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \right) z^{k-1} + P_d(z) \mathbf{D}_d \right) \mathbf{u} \quad (94)$$

$$= \mathbf{G}_d(z) \mathbf{u}. \quad (95)$$

Note that since  $\mathbf{x}_0 = \mathbf{0}$  and  $\mathbf{u}[n]$  and  $\mathbf{y}_d[n]$  vanish for  $n < 0$ , the state equations (26-27) hold for all  $n \in \mathbb{Z}$  and, therefore, (11) also holds for all  $n \in \mathbb{Z}$ .

We now prove the second part of the theorem that states that it is possible to decode the input sequence from the output if and only if the matrix  $\mathbf{G}_d(z)$  is of full column rank over the polynomial ring  $\mathbb{F}[z]$ . If it is not, there exists a vector of polynomials  $\mathbf{v}_d(z)$  such that

$$\mathbf{G}_d(z) \mathbf{v}_d(z) = \mathbf{0}. \quad (96)$$

Denote the maximal degree of the polynomials in  $\mathbf{v}_d(z)$  by  $\delta$ . Let  $\mathbf{u}_{d,0}[n]$  be the sequence defined by  $\mathbf{u}_{d,0}[n] = (\mathbf{v}_d(z)\psi)[n]$ , where the sequence  $\psi[n]$  is

$$\psi[n] = \begin{cases} 1 & \text{if } n = \delta \\ 0 & \text{if } n \neq \delta \end{cases}. \quad (97)$$

Note that  $\mathbf{u}_{d,0}[n]$  vanishes for  $n < 0$  and, therefore,  $\mathbf{u}_{d,0}$  is a legal input sequence that will lead to a zero sequence  $\mathbf{G}_d(z)\mathbf{u}_{d,0}$ . In view of (11), this will lead to a zero sequence  $P_d(z)\mathbf{y}_d$ . We assumed that  $P_d(z)$  is not the zero polynomial and, hence, the output sequence  $\mathbf{y}_d$  will also vanish, since the relationship between  $\mathbf{y}_d$  and  $P_d(z)\mathbf{y}_d$  is injective. In that case, no decoding method will tell if the input sequence was  $\mathbf{u}_{d,0}[n]$  or a totally zero sequence.

On the other hand, if  $\mathbf{G}_d(z)$  is of full column rank then we can show that the input sequence can be decoded from the output sequence. We apply a decoding scheme that is slightly different to the sequential decoder [7]. We multiply both sides of (11) by  $\text{adj}(\mathbf{G}_d(z))$  (we can assume that  $\mathbf{G}_d(z)$  is a square matrix, since, if not, we can remove some of its linearly dependent rows to make it square) to get

$$\mathbf{q} := P_d(z)\mathbf{y}_d \quad (98)$$

$$\mathbf{w} := \text{adj}(\mathbf{G}_d(z))\mathbf{q} \quad (99)$$

$$\stackrel{(a)}{=} \text{adj}(\mathbf{G}_d(z))\mathbf{G}_d(z)\mathbf{u}$$

$$\stackrel{(b)}{=} \det(\mathbf{G}_d(z))\mathbf{u}$$

$$= f_d(z)\mathbf{u},$$

where

(a) follows from (11),

(b) follows from the fact that for any square matrix  $\mathbf{G}$  over the polynomial ring  $\mathbb{F}[z]$ , the following identity holds:

$$\text{adj}(\mathbf{G})\mathbf{G} = \det(\mathbf{G})\mathbf{I},$$

where  $\mathbf{I}$  denotes the identity matrix. The polynomial  $f_d(z) := \det(\mathbf{G}_d(z))$  is a non zero polynomial, since we assumed that  $\mathbf{G}_d(z)$  is of full column rank.

$$f_d(z) = \sum_{i=0}^k \alpha_i z^i, \quad \alpha_k \neq 0.$$

If we know the sequence  $\mathbf{y}_d[n]$  we can compute  $\mathbf{w}[n]$  and from that find  $\mathbf{u}[n]$ :

$$\mathbf{w}[n] = \sum_{i=0}^k \alpha_i \mathbf{u}[n+i], \quad (100)$$

$$\mathbf{u}[n] = (\alpha_k)^{-1} \left( \mathbf{w}[n-k] - \sum_{i=0}^{k-1} \alpha_i \mathbf{u}[n-k+i] \right). \quad (101)$$

■

*Proof for Theorem 3:* We first prove that if  $P_d(t)$  satisfies (43)-(44), then its coefficients are a solution of (18). We substitute the expression for  $\mathbf{y}_d[n]$  from (42) into (18) to get:

$$\sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j+\tau] = \quad (102)$$

$$\begin{aligned} &= \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau} \mathbf{x}_0 + \sum_{j=0}^N \sum_{k=1}^{\min\{m, \lfloor \frac{j+\tau-1}{2N+1} \rfloor\}} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau-1-(2N+1)k} \mathbf{B} \mathbf{e}_k \\ &+ \sum_{j=0}^N \sum_{k=1}^m \alpha_{d,j} \mathbf{D}_d \mathbf{e}_k 1_{\{j+\tau=(2N+1)k\}} \end{aligned} \quad (103)$$

$$\begin{aligned} &\stackrel{(a)}{=} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^\tau \mathbf{x}_0 + \sum_{j=0}^N \sum_{k=1}^{\lfloor \tau/(2N+1) \rfloor} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau-1-(2N+1)k} \mathbf{B} \mathbf{e}_k \\ &+ \sum_{j=0}^N \sum_{k=1}^m \alpha_{d,j} \mathbf{D}_d \mathbf{e}_k 1_{\{j+\tau=(2N+1)k\}} \end{aligned} \quad (104)$$

$$\stackrel{(b)}{=} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^\tau \mathbf{x}_0 + \sum_{k=1}^{\lfloor \tau/(2N+1) \rfloor} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1-(2N+1)k} \mathbf{B} \mathbf{e}_k + \mathbf{0} \quad (105)$$

$$\stackrel{(c)}{=} \mathbf{0}, \quad (106)$$

where

(a) follows from the fact that  $0 \leq j \leq N$  and that

$$\tau = (2N+1)p + \tilde{\tau}, \quad \text{where } 0 \leq p \leq m, \quad 1 \leq \tilde{\tau} \leq N, \quad (107)$$

and therefore

$$\lfloor \frac{j+\tau-1}{2N+1} \rfloor = \lfloor p + \frac{j+\tilde{\tau}-1}{2N+1} \rfloor = p, \quad (108)$$

(b) follows from the fact that  $j+\tau$  cannot be a multiple of  $(2N+1)$ ,

(c) follows from (43)-(44).

We now prove the veracity of Theorem 3 in the reverse direction. We assume the coefficients  $\{\alpha_{d,j}\}_{j=0}^N$  of the polynomial  $P_d(t) = \sum_{j=0}^N \alpha_{d,j} t^j$  satisfy (18) and show that  $P_d(t)$  satisfies (43)-(44). We first note that

$$\mathbf{y}_d[n] = \mathbf{C}_d \mathbf{A}^n \mathbf{x}_0 \quad \forall 0 \leq n \leq 2N, \quad (109)$$

so if (18) is satisfied for  $\tau \in \{1, \dots, N\}$ , then (44) is also satisfied for  $\tau \in \{0, \dots, N-1\}$  and, hence, for all  $\tau \geq 0$  (by Lemma 3). For  $\tau \in \{(2N+1)+1, \dots, (2N+1)+N\}$ , we have:

$$\mathbf{0} = \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + \tau] \quad (110)$$

$$= \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau-1-(2N+1)} \mathbf{B} \mathbf{e}_1 + \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau} \mathbf{x}_0 \quad (111)$$

$$\stackrel{(a)}{=} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1-(2N+1)} \mathbf{B} \mathbf{e}_1 + \mathbf{0}, \quad (112)$$

where (a) is because (44) holds. In view of Lemma 3, we see that (113) holds for  $k = 1$ .

$$\mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^\tau \mathbf{B} \mathbf{e}_k = \mathbf{0}, \quad \forall \tau \geq 0. \quad (113)$$

By induction on  $k$ , we can prove that (113) holds for all  $k \in \{1, \dots, m\}$ . For all  $\tau \in \{(2N+1)k+1, \dots, (2N+1)k+N\}$ , we have:

$$\mathbf{0} = \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + \tau] \quad (114)$$

$$= \sum_{k'=1}^k \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau-1-(2N+1)k'} \mathbf{B} \mathbf{e}_{k'} + \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j+\tau} \mathbf{x}_0 \quad (115)$$

$$= \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1-(2N+1)k} \mathbf{B} \mathbf{e}_k + \sum_{k'=1}^{k-1} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1-(2N+1)k'} \mathbf{B} \mathbf{e}_{k'} + \mathbf{0} \quad (116)$$

$$\stackrel{(a)}{=} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{\tau-1-(2N+1)k} \mathbf{B} \mathbf{e}_k + \mathbf{0}, \quad (117)$$

where (a) follows from the induction assumption of (113) on  $k' < k$ . In view of Lemma 3, we see that (113) holds for all  $0 \leq k \leq m$  and, therefore, (43) holds as well.  $\blacksquare$

*Proof for Theorem 4:* First we note that the polynomial  $P_d(z) = \sum_{k=0}^N \alpha_{d,k} z^k$  found in Algorithm 2 satisfies (43)-(44) and the transfer matrix  $\mathbf{G}_d(z)$  can be described as follows:

$$\mathbf{G}_d(z) = [\mathbf{g}_{d,1}(z), \mathbf{g}_{d,2}(z), \dots, \mathbf{g}_{d,m}(z)], \quad (118)$$

$$\mathbf{g}_{d,i}(z) = \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + (2N+1)i - k + 1] z^{k-1}, \quad \forall i \in \{1, \dots, m\}, \quad (119)$$

where  $\{\mathbf{y}_d[n]\}$  are defined in (42). To prove the theorem, first we find an expression for  $\mathbf{g}_{d,i}(z)$  by substituting (42) into (119), and we show that  $\mathbf{G}_d(z)$  is given by (85). Then we explain why (45) holds despite the fact that  $\mathbf{x}_0 \neq \mathbf{0}$ .

$$\begin{aligned} \mathbf{g}_{d,i}(z) &= \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \mathbf{y}_d[j + (2N+1)i - k + 1] z^{k-1} \\ &= \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^j \mathbf{A}^{(2N+1)i-k+1} \mathbf{x}_0 z^{k-1} \\ &\quad + \sum_{k=1}^{N+1} \sum_{j=0}^N \sum_{k'=0}^{\min\{m, \lfloor \frac{j-k}{2N+1} + i \rfloor\}} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k+(2N+1)(i-k')} \mathbf{B} \mathbf{e}_{k'} z^{k-1} \\ &\quad + \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \sum_{k'=1}^m \mathbf{D}_d \mathbf{e}_{k'} 1_{\{j+(2N+1)i-k+1=(2N+1)k'\}} z^{k-1}. \end{aligned} \quad (120)$$

The first part of (120) vanishes because of (44)

$$\begin{aligned} \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^j \mathbf{A}^{(2N+1)i-k+1} \mathbf{x}_0 z^{k-1} &= \sum_{k=1}^{N+1} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{(2N+1)i-k+1} \mathbf{x}_0 z^{k-1} \\ &= \mathbf{0}. \end{aligned}$$

The second part of (120) can be written as:

$$\begin{aligned} &\sum_{k=1}^{N+1} \sum_{j=0}^N \sum_{k'=0}^{\min\{m, \lfloor \frac{j-k}{2N+1} + i \rfloor\}} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k+(2N+1)(i-k')} \mathbf{B} \mathbf{e}_{k'} z^{k-1} \\ &\stackrel{(a)}{=} \sum_{k=1}^N \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \mathbf{e}_i z^{k-1} + \sum_{k=1}^{N+1} \sum_{j=0}^N \sum_{k'=0}^{i-1} \alpha_{d,j} \mathbf{C}_d \mathbf{A}^j \mathbf{A}^{(2N+1)(i-k')-k} \mathbf{B} \mathbf{e}_{k'} z^{k-1} \\ &= \sum_{k=1}^N \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \mathbf{e}_i z^{k-1} + \sum_{k=1}^{N+1} \sum_{k'=0}^{i-1} \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^{(2N+1)(i-k')-k} \mathbf{B} \mathbf{e}_{k'} z^{k-1} \end{aligned}$$



$$\stackrel{(b)}{=} \sum_{k=1}^N \sum_{j=k}^N \alpha_{d,j} \mathbf{C}_d \mathbf{A}^{j-k} \mathbf{B} \mathbf{e}_i z^{k-1}, \quad (121)$$

where

(a) is obtained by splitting the summation over  $k'$  from 0 to  $i-1$ , and for  $k' = i$  (only when  $k \leq j$ ).

(b) holds because  $P_d(z)$  satisfies (43).

The third part of (120) is:

$$\begin{aligned} & \sum_{k=1}^{N+1} \sum_{j=0}^N \alpha_{d,j} \sum_{k'=1}^m \mathbf{D}_d \mathbf{e}_{k'} 1_{\{j+(2N+1)(i-k')-k+1=0\}} z^{k-1} \\ &= \sum_{j=0}^N \alpha_{d,j} z^j \mathbf{D}_d \mathbf{e}_i \\ &= P_d(z) \mathbf{D}_d \mathbf{e}_i. \end{aligned} \quad (122)$$

By combining all  $\mathbf{g}_{d,i}$  vectors into a matrix we get that  $\mathbf{G}_d(z)$  is given by (85). It was already proved in Theorem 2 that for such a  $\mathbf{G}_d(z)$  matrix, the difference equation (11) holds between  $\mathbf{u}$  and  $\mathbf{y}_d$ , provided that  $\mathbf{x}_0 = \mathbf{0}$ . For cases in which the initial state is not zero, the output sequence  $\mathbf{y}_d[n]$  can be written as a sum of the zero input response  $\mathbf{y}_{\text{ZIR},d}[n]$  and the zero state response  $\mathbf{y}_{\text{ZSR},d}[n]$  sequences, where:

$$\mathbf{y}_{\text{ZIR},d}[n] = \mathbf{C}_d \mathbf{A}^n \mathbf{x}_0, \quad (123)$$

and  $\mathbf{y}_{\text{ZSR},d}[n]$  is the output of the network, as if the initial state were zero. Finally, using Theorem 2 for  $\mathbf{y}_{\text{ZSR},d}[n]$  and (44) for  $\mathbf{y}_{\text{ZIR},d}[n]$  we get for all  $n \geq 1$ :

$$(P_d(z) \mathbf{y}_d)[n] = (P_d(z) \mathbf{y}_{\text{ZIR},d})[n] + (P_d(z) \mathbf{y}_{\text{ZSR},d})[n] \quad (124)$$

$$= \mathbf{C}_d P_d(\mathbf{A}) \mathbf{A}^n \mathbf{x}_0 + (\mathbf{G}_d(z) \mathbf{u})[n] \quad (125)$$

$$= (\mathbf{G}_d(z) \mathbf{u})[n]. \quad (126)$$

■

*Proof for Theorem 5:* First, we show that if for every  $s \in \mathcal{S}$  there are  $R'_s$  linearly independent column vectors  $\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}$  from the columns of the matrix  $\mathbf{G}_{d,s}(z)$ , such that  $\cup_{s \in \mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$  is a set of linearly independent vectors, then the rates  $(R'_s)_{s \in \mathcal{S}}$  are

achievable for the sink node  $d$  with the current local encoding kernels. Every source node  $s$  can transmit its input symbols only on the inputs  $u_{s,i}$  that correspond to the vectors  $\{\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}\}$  and zeros on the other inputs:

$$u_{s,i} = \begin{cases} \text{The zero sequence} & \text{if column } i \text{ of } \mathbf{G}_{d,s}(z) \notin \{\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}\} \\ \text{A non zero sequence} & \text{if column } i \text{ of } \mathbf{G}_{d,s}(z) \in \{\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}\} \end{cases}. \quad (127)$$

In that case (11) can be simplified into

$$P_d(z)\mathbf{y}_d = \tilde{\mathbf{G}}_d(z)\tilde{\mathbf{u}}, \quad (128)$$

where  $\tilde{\mathbf{u}}[n]$  is the input sequence with all zero inputs removed and  $\tilde{\mathbf{G}}_d(z)$  is the matrix

$$\tilde{\mathbf{G}}_d(z) = \begin{bmatrix} \mathbf{v}_{s_1,1}, \dots, \mathbf{v}_{s_1,R'_{s_1}} & \mathbf{v}_{s_2,1}, \dots, \mathbf{v}_{s_2,R'_{s_2}} & \dots & \mathbf{v}_{s_{|\mathcal{S}|},1}, \dots, \mathbf{v}_{s_{|\mathcal{S}|},R'_{s_{|\mathcal{S}|}}} \end{bmatrix}. \quad (129)$$

From the proof of the second part of Theorem 2, we know that it is possible to decode  $\tilde{\mathbf{u}}$  from  $\mathbf{y}_d$  if and only if the matrix  $\tilde{\mathbf{G}}_d(z)$  is of full column rank, i.e. all of its column vectors are linearly independent over  $\mathbb{F}[z]$ . We assumed that the columns of  $\tilde{\mathbf{G}}_d(z)$  are linearly independent and, therefore, the rates  $(R'_s)_{s \in \mathcal{S}}$  are achievable for the sink node  $d$ .

We now prove the second part of Theorem 5. We assume that the rates  $(R'_s)_{s \in \mathcal{S}}$  are achievable for a sink node  $d$  and we show that for every  $s \in \mathcal{S}$  there are  $R'_s$  linearly independent column vectors  $\mathbf{v}_{s,1}, \dots, \mathbf{v}_{s,R'_s}$  from the columns of the matrix  $\mathbf{G}_{d,s}(z)$ , such that the vectors  $\cup_{s \in \mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$  are linearly independent. By the definition of achievable rates, we know that every source  $s \in \mathcal{S}$  can transmit zeros on  $(R_s - R'_s)$  of its input sequences, such that  $d$  will be able to decode  $\mathbf{u}$  from  $\mathbf{y}_d$ . If that is the case, the relationship between the input and the output sequences is described by (128), where  $\tilde{\mathbf{u}}[n]$  is the input sequence with all zero inputs removed, and  $\tilde{\mathbf{G}}_d(z)$  is the matrix  $\mathbf{G}_d(z)$  with removed column vectors that correspond to the zero input sequences. Since  $\tilde{\mathbf{u}}$  is decodable, we know that the column vectors of  $\tilde{\mathbf{G}}_d(z)$  are linearly independent over  $\mathbb{F}[z]$ . The  $R'_s$  column vectors of  $\tilde{\mathbf{G}}_d(z)$  that correspond to the non zero inputs of a source node  $s$  are also column vectors of  $\mathbf{G}_{d,s}(z)$ , since  $\mathbf{G}_{d,s}(z)$  contains all the column vectors of  $\mathbf{G}_d(z)$  that correspond to the input sequence  $\mathbf{u}_s$ . Therefore, we showed that for every source  $s$ , there are  $R'_s$  linearly independent vectors from the columns of  $\mathbf{G}_{d,s}(z)$ , such that all the vectors together are linearly independent. This completes the proof.  $\blacksquare$

## ACKNOWLEDGMENTS

The authors would like to thank Dr. Izchak Lewkowicz for his lectures in Linear Control theory, which were of great help.

## REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, 2003.
- [3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, 2003.
- [4] S.-Y. Li and R. W. Yeung, "On convolutional network coding," in *Proc. IEEE Int. Symp. Inform. Theory*, 2006. IEEE, pp. 1743–1747.
- [5] S.-Y. Li and Q. T. Sun, "Network coding theory via commutative algebra," *IEEE Trans. Inform. Theory*, vol. 57, no. 1, pp. 403–415, 2011.
- [6] E. Erez and M. Feder, "Efficient network codes for cyclic networks," in *Proc. IEEE Int. Symp. Inform. Theory*, 2005. IEEE, pp. 1982–1986.
- [7] E. Erez and M. Feder, "Efficient network code design for cyclic networks," *IEEE Trans. Inform. Theory*, pp. 3862–3878, 2010.
- [8] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
- [9] R. W. Yeung, *Information theory and network coding*. Springer, 2008.
- [10] W. Guo, X. Shi, N. Cai, and M. Médard, "Localized dimension growth: a convolutional random network coding approach to managing memory and decoding delay," 2013.
- [11] W. Guo, N. Cai, and Q. T. Sun, "Time-variant decoding of convolutional network codes," *Communications Letters*, pp. 1656–1659, 2012.
- [12] T. Katayama, *Subspace methods for system identification*. Springer, 2006.
- [13] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proc. IEEE Int. Conf. Communications*, 2004, vol. 2. IEEE, pp. 661–666.
- [14] W. J. Rugh, *Linear system theory*. Prentice-Hall, Inc., 1996.
- [15] V. Sreeram and A. Zomaya, "Note on the hankel matrix," *Electronics Letters*, vol. 27, no. 4, pp. 362–364, 1991.
- [16] N. Cai and W. Guo, "The conditions to determine convolutional network coding on matrix representation," in *Proc. NetCod*, 2009. IEEE, pp. 24–29.
- [17] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra, 2nd edition*. Prentice-Hall, 1997.